REMARKS

Claims 1-38 are currently pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Claim Objection:

The Examiner objects to claims 1-38 and asserts that the claims use the term "thread" to mean "a queue". Applicants traverse the Examiner's objection. Claims 1-38 do not use the term "thread" to mean a "queue." In fact, only claims 25 and 37 recite the term "thread" and the term "thread" is not used in claims 25 and 37 to mean "a queue" as asserted by the Examiner. For example, claim 25 refers to a primary scheduler executed in a single thread and a secondary scheduler executed in a different thread. Claim 25 is not referring to a primary scheduler executing in a "queue", as the Examiner contends, but rather refers to a primary scheduler executing in a "thread", as would be understood by anyone of ordinary skill in the art. Others of the claims use to the terms, "thread-safe", "multi-threaded", "single-threaded", which are also not used to mean "a queue", as suggested by the Examiner.

The Examiner also states, "[i]n claims 12-13 the requests recited are queues which are being held by pending requests from schedulers." The Examiner appears to have misread claims 12-13. For example, claim 12, recites, "[t]he system of claim 1, wherein the *primary scheduler comprises a primary queue* which is operable to *hold pending requests and responses* to the requests" (emphasis added). Claim 12 does not recite that requests are queues, as the Examiner contends. Nor does claim 12 recite queues that are held by pending requests as asserted by the Examiner. Claim 12 clearly refers to a primary queue operable to hold pending requests and responses to the requests. The requests recited in claim 12 are not queues, nor is there anything in the claims that suggests they are. Similarly, claim 13 refers to a secondary queue operable to hold pending requests.

Furthermore, the Examiner also contends (erroneously) that "[t]he thread recited in claim 1 is the same as the queue recited in claims 12-13". However, claim 1 does not recite a "thread". In contrast, claim 1 refers to a primary schedule executable to schedule requests for networked data resources and a secondary scheduler executable to receive requests from a multi-threaded application in a thread-safe manner and to send the requests to the primary scheduler in a thread-safe manner. As noted above, claim 12 refers to a primary queue comprised in the primary scheduler.

Thus, claims 1-38 use the term "thread" when referring to a thread, and the term "queue" when referring to a queue. The Examiner has clearly misunderstood and misconstrued the language of applicants' claims. A queue may be implemented by one or more threads, but Applicants' claims do not equivocate a thread to a queue.

Section 112, Second Paragraph, Rejection:

The Office Action rejected claims 1-38 under 35 U.S.C. § 112, second paragraph as indefinite. Applicants respectfully traverse this rejection for at least the following reasons.

Regarding claim 15, the Examiner contends that the term "manager application" lacks antecedent basis. However, claim 15 recites, in part, "receiving a plurality of management requests from a multi-threaded manager application" (emphasis added). The subsequent reference to "the manager application" clearly refers to "a multi-threaded manager application" introduced previously since no other manager application is mentioned in the claim. The term "multi-threaded" is simply an adjective describing the manger application, not an indication of a separate manager application. The reference to "the manager application" at lines 6-7 of claim 15 clearly refers to the same manager application as introduced at lines 3-4 of claim 15.

Regarding claim 6, the Examiner asserts that the term "resources" lacks proper antecedent basis. However, claim 6 depends from claim 1 and claim 1 refers to a primary

scheduler executable to schedule requests for networked data *resources*. As there are no other references to resources, the term "the resources" in claim 6 clearly refers to the networked data resources of claim 1.

Regarding claim 1, the Examiner contends that the term "thread-safe manner" is unclear. Applicants disagree. The terms "thread-safe" and "in a thread-safe manner" are terms of art well understood by anyone of ordinary skill in the art.

The Examiner also argues that it is unclear "which requests are being sent to the primary scheduler". However, claim 1 clearly refers to a secondary scheduler executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests (received from the multi-threaded application) to the primary scheduler in a thread-safe manner. Thus, applicants submit that claim 1 clearly describes which requests are being sent to the primary scheduler.

The Examiner further contends that it is unclear "how the secondary scheduler would access the networked date [sic] resource." However, the Examiner has again mischaracterized Applicants' claim. Claim 1 does not recite anything regarding the secondary schedule accessing the network data resources, as the Examiner (erroneously) contends. Instead, claim 1 refers to a secondary scheduler executable to receive a plurality of requests. Thus, the Examiner's concern over the clarity of how the secondary scheduler would access networked data resources makes no sense in light of the language of claim 1. Other than the limitations recited in the claim, claim 1 does not limit how the secondary scheduler receives the requests. Applicants remind the Examiner that breadth of a claim is not to be equated with indefiniteness. M.P.E.P. 2173.04.

Similar remarks apply as those above regarding claim 1 also to the Examiner's objections to claims 15 and 27.

Regarding claim 6, the Examiner argues, "it is unclear which resources are being used (i.e. the networked data resources or new resources)" (parentheses by Examiner).

However, claim 6 refers to "the resources" and, as noted above, the only resources recited by claim 6 or claim 1, from which claim 6 depends, are the networked data resources. As claim 6 refers to "the resources" it is quite clear that claim 6 is referring to the networked data resources recited in claim 1.

The Examiner also contends, "it is unclear which requests are being sent, (i.e. requests from the primary or secondary scheduler?)" (parentheses by Examiner). However, neither claim 6, nor claim 1, refers to any requests sent *from* the primary scheduler. Claim 6 makes no reference to the sending of any requests, but instead recites the limitation "wherein the requests comprise management requests". Claim 1 refers to a secondary scheduler executable to receive requests *from a multi-threaded application* and to send the requests *to the primary scheduler*. Thus, applicants assert that the language of claim 6 is quite clear regarding resources and requests.

Similar remarks as those above regarding claim 6 also apply to the Examiner's objection to claim 7 as well.

Regarding claim 15, the Examiner asserts, "is it unclear whether the managed objects (lines 11) are the same managed objects on a network (lines 1-2)" (parentheses by Examiner). As there is only one set of managed objects referred to by claim 15, and since proper antecedent basis is used, applicants assert that claim 15 is not unclear regarding the managed objects.

The Examiner also argues regarding claim 15 that "it is unclear what the determining process is as to which management requests to send to the primary scheduler (i.e. do all the management requests go to the primary scheduler or just some)" (parentheses by Examiner). Applicants can find no basis for the Examiner's objection as claim 15 does not refer to, or recite, any such "determining process". Instead, claim 15 recites, in part, "sending the management requests from the secondary scheduler to a primary scheduler in a thread-safe manner". Thus, "the management requests" clearly

refers to the plurality of management requests received from a multi-threaded manager application into a secondary scheduler recited previously in claim 15.

The Examiner further contends, "it is unclear how the primary scheduler can access the managed objects (i.e. there is no connection between them)" (parentheses by Examiner). Claim 15 recites, in part, "executing the management requests on the managed objects". Claim 15 is not limited to any particular connection between the primary scheduler and the managed objects, nor is there any requirement that it should be so limited. (For example, claim 16 recites a further limitation on executing the management requests on the managed objects comprising sending the management requests to a management information server coupled to the managed objects.)

Similar remarks as those above regarding claim 15 also apply the Examiner's objection to claim 27.

Applicants further remind the Examiner that if the "scope of a claim would be reasonably ascertained by those skilled in the art, then the claim is not indefinite" (M.P.E.P. § 2173.05(e)). As anyone skilled in the art would be able to reasonably ascertain the scope of Applicants' claims, Applicants assert that claims 1- 38 are not indefinite.

Section 103(a) Rejection:

The Office Action rejected claims 1-38 under 35 U.S.C. § 103(a) as being unpatentable over Kimmel et al. (U.S. Patent 6,105,053) (hereinafter "Kimmel") and further in view of Maresco (U.S. Patent 6,418,458). Applicants respectfully traverse the rejection of claims 1-38 for at least the following reasons.

Regarding claim 1, contrary to the Examiner's assertion, Kimmel in view of Maresco fails to teach or suggest a primary scheduler which is executable to schedule requests for networked data resources. The Examiner cites column 24, lines

14-26 and column 6, lines 43-65 of Kimmel. Kimmel teaches an operating system for non-uniform memory access (NUMA) multiprocessor systems and utilizes a software abstraction of the system hardware to maintain balanced processor and memory loads (Kimmel, Abstract). The Examiner's first cited passage (column 24, lines 14-26) describes how Kimmel's operating system may function in systems having various numbers of execution queue levels and that have differently sized thread groups. The second passage cited by the Examiner describes the thread group structure maintaining cumulative timeslice and job processor (JP) accounting for all threads in a thread group. This cited passage also describes how a JP dispatcher selects a thread group to execute. Kimmel teaches that a dispatcher selects an individual thread from a thread group based on the local priority and scheduling policy. Selection occurs at two levels. Global scheduling policies are used to select a thread group, while local scheduling policies are used to select an individual thread from the selected thread group.

Neither of the Examiner's cited passages teaches or suggests a primary scheduler executable to schedule *requests for networked data resources*. Kimmel's JP dispatcher selects a thread to execute on a processor of the NUMA system. Kimmel fails to mention any networked data resources. All resources taught by Kimmel are local to a single NUMA system and are not networked data resources. Additionally, Kimmel makes no mention of *requests* for networked data resources. Kimmel states that the "dispatcher is a kernel subsystem that is a mechanism responsible for scheduling and executing processes on an associated JP [job processor] in accordance with certain global and local scheduling policies" (Kimmel, column 5, lines 43-46). Kimmel is concerned with balanced processor and memory loads in a NUMA system. Kimmel is not concerned with *scheduling requests for networked data resources*.

Maresco pertains to the creation of threads to prioritize the execution of tasks in an object oriented system (Maresco, column 1, lines 42-57; column 2, lines 25-38). Neither Kimmel nor Maresco has anything to do with scheduling requests for networked data resources. All resources taught by Kimmel are local to a single NUMA system and are not networked data resources. And the threads in Maresco are clearly not requests for

networked data resources. Thus, Kimmel and Maresco clearly do not teach or suggest anything about scheduling requests for networked data resources.

In further regard to claim 1, contrary to the Examiner's assertion, Kimmel in view of Maresco also fails to teach or suggest a secondary scheduler, wherein the secondary scheduler is executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests to the primary scheduler in a thread-safe manner. The Examiner cites several passages of Kimmel referring to Kimmel's medium term scheduler. However, Kimmel's medium term scheduler does not receive any requests from a multi-threaded application. Instead, Kimmel's medium term scheduler "monitors the progress of active processes in the system and sets a flag for those processes that are not progressing" (Kimmel, column 2, lines 40-45). Nowhere does Kimmel describe his medium term scheduler receiving requests from a multi-threaded application. Furthermore, the medium term scheduler in Kimmel's system does not send requests received from a multi-threaded application to Kimmel's dispatcher, which the Examiner equates to the primary scheduler of claim 1. The Examiner has not pointed out any piece or feature of Kimmel's system that can be interpreted as a secondary scheduler executable to *receive requests from a multi-threaded application* and to *send the requests to a primary scheduler*.

Instead, Kimmel's medium term scheduler monitors thread groups to identify languishing thread groups and candidates for load balancing by monitoring the loads of the respective scheduling locals (i.e. certain nodes in Kimmel's hierarchical representation of a system's hardware) to identify any load imbalances and by identifying any thread groups that do not have the same current and home scheduling locales (Kimmel, column 10, lines 34-40). As described at one of Examiner's cited passages (Kimmel, column 10, lines 50-65), when Kimmel's medium term scheduler identifies a languishing thread group, it may do one of three things. First it may raise the thread group's priority. Secondly it may assign the thread group to the run queue of a higher node. And thirdly the medium term scheduler may set a flag associated with the thread group to allow poaching of the thread group by another JP. Thus, Kimmel's medium

term scheduler does not receive requests from a multi-threaded application. Nor does Kimmel's medium term scheduler send the requests to the JP dispatcher, which the Examiner equates to a primary scheduler. Instead, as noted above, the medium term schedule monitors the state of various thread groups through Kimmel's thread group structure and changes priorities and scheduling locales accordingly to maintain balanced processor and memory loads.

Maresco is relied upon by the Examiner only to teach "a thread safe system" and cites column 2, lines 30-38. However, Maresco fails to overcome any of the above noted deficiencies in Kimmel regarding claim 1. Thus, the Examiner's combination of Kimmel in view of Maresco would not result in a system that includes a primary scheduler which is executable to schedule requests for networked data resources; and a secondary scheduler, wherein the secondary scheduler is executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests to the primary scheduler in a thread-safe manner. Instead, the Examiner's proposed combination of Kimmel and Maresco would result in Kimmel's system for maintaining balanced processor and memory loads in a NUMA system where Kimmel's thread groups are the work crews of Maresco.

Moreover, Maresco does not teach or suggest anything about receiving or sending requests for networked data resources in a thread-safe manner. Instead, Maresco is concerned with safely *creating* threads for task execution is a computer system (Maresco, column 1, lines 42-57; column 2, lines 25-38). Maresco teaches nothing about a scheduler receiving and sending requests in a thread-safe manner. Just because Maresco mentions thread creation for task execution does not imply that Maresco suggests receiving and sending requests by schedulers in a thread-safe manner. As discussed above, neither Kimmel nor Maresco has anything to do with scheduling requests from a multi-threaded application, let alone receiving and sending requests for scheduling in a thread-safe manner.

For at least the reasons above, the rejection of claim 1 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 15, contrary to the Examiner's assertion, Kimmel in view of Maresco fails to teach or suggest receiving a plurality of management requests from a multi-threaded manager application into a secondary scheduler in a thread-safe manner. The Examiner fails to cite any portion of Kimmel or Maresco that teaches receiving a plurality of management requests from a multi-threaded manager application in the rejection of claim 15. Furthermore, as noted above regarding claim 1, neither Kimmel nor Maresco describes anything regarding receiving management requests from a multi-threaded manager application. Neither Kimmel nor Maresco are concerned with management requests from multi-threaded manager applications. Instead, Kimmel and Maresco teach respective systems for managing the execution of threads within a multi-processor computer system. For a more detailed discussion regarding Kimmel's and Maresco's failure to teach or suggest receiving requests (whether management requests or any other requests) from a multi-threaded application (whether a manager application or any other application) please the remarks above regarding claim 1.

Additionally, Kimmel in view of Maresco fails to teach or suggest scheduling the plurality of management requests in a secondary queue in the secondary scheduler after receiving the management requests from the manager application. The Examiner cites several passages of Kimmel referring to Kimmel's medium term scheduler. However, as discussed above, Kimmel's medium term scheduler does not receive any requests from a multi-threaded application and clearly does not receive management requests from a multi-threaded manager application. Instead, as noted above, Kimmel's medium term scheduler "monitors the progress of active processes in the system and sets a flag for those processes that are not progressing" (Kimmel, column 2, lines 40-45). Furthermore, Kimmel's medium term scheduler does not schedule management requests, but instead monitors and adjusts the priority of threads in thread groups executing on a multi-processor system.

In further regard of claim 15, Kimmel in view of Maresco also fails to teach or suggest sending the management requests from the secondary scheduler to a primary scheduler in a thread-safe manner. As noted above regarding claim 1, Kimmel fails to describe his medium term scheduler, which the Examiner equates to the secondary scheduler, as sending management requests to his dispatcher, which the Examiner equates with a primary scheduler. Instead, both the medium term scheduler and the dispatcher access Kimmel's hierarchical node structure to monitor the states of the individual threads and thread groups, adjusting the priority and scheduling policies to ensure balanced processor and memory loading. Nowhere does Kimmel describe his medium term scheduler sending management requests to the dispatcher. Maresco also fails to mention anything regarding sending management requests from a secondary scheduler to a primary scheduler. For a more detailed discussion regarding Kimmel's and Maresco's failure to teach or suggesting sending management requests from a secondary scheduler to a primary scheduler, please refer to the discussion above regarding claim 1.

Furthermore, Kimmel in view of Maresco fails to teach or suggest executing the management requests on the managed objects after scheduling the management requests in the primary queue. The Examiner cites column 6, lines 62-67 where Kimmel describes how using thread groups in developing processes give a user flexibility "to choose between creating a new thread within an existing thread group or creating a new thread group." The cited passage makes no mention of executing management requests on managed objects. Creating thread groups and giving users the ability to choose between adding a new thread to an existing thread group or creating a new thread group has no relevance to management requests or managed objects, as they are understood in the art. As noted above, nowhere does Kimmel mention anything regarding either management requests for managed objects. Maresco is only relied upon the by the Examiner to teach a thread safe system. Maresco fails to overcome any of the above noted deficiencies of Kimmel.

Thus, the Examiner's proposed combination of Kimmel and Maresco fails to teach or suggest receiving a plurality of management requests from a multi-threaded manager application into a secondary scheduler in a thread-safe manner; scheduling the plurality of management requests in a secondary queue in the secondary scheduler after receiving the management requests from the manager application; sending the management requests from the secondary scheduler to a primary scheduler in a thread-safe manner; or executing the management requests on the managed objects after scheduling the management requests in the primary queue. Instead, as noted above regarding claim 1, the Examiner's proposed combination of Kimmel and Maresco would result in Kimmel's system for maintaining balanced processor and memory loads in a NUMA system where Kimmel's thread groups are the work crews of Maresco.

For at least the reason given above, the rejection of claim 15 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 15 also apply to claim 27.

Regarding claim 2, Kimmel in view of Maresco fails to teach or suggest, contrary to the Examiner's assertion, wherein the primary scheduler is single-threaded. The Examiner cites column 5, lines 50-56 and column 24, lines 14-26. However, both of the Examiner's cited passages describe how Kimmel's system may be used in conjunction with thread groups including various numbers of threads, including single thread sized thread groups. The cited passages do not mention anything regarding whether Kimmel's dispatcher, which the Examiner equates to a primary scheduler, is single-threaded. Additionally, Maresco fails to teach anything regarding wherein a primary scheduler is single-threaded and thus fails to overcome the above noted deficiency of Kimmel. Thus, the rejection of claim 2 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 3, Kimmel in view of Maresco fails to teach or suggest wherein the secondary scheduler is multi-threaded, in contrast to the Examiner's contention. The

Examiner again cites column 5, lines 50-56 and column 24, lines 14-26. However, as noted above, both of the Examiner's cited passages describe how Kimmel's system may be used in conjunction with thread groups including various numbers of threads, including single thread sized thread groups. Neither of the cited passages mention anything regarding whether Kimmel's medium term scheduler, which the Examiner equates to a secondary scheduler, is multi-threaded. Additionally, Maresco fails to teach anything regarding wherein a secondary scheduler is multi-threaded and thus fails to overcome the above noted deficiency of Kimmel. Thus, the rejection of claim 3 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 4, Kimmel in view of Maresco fails to teach or suggest wherein secondary scheduler is executable to receive the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. However, as noted above regarding claim 1, Kimmel (and Maresco) fails to teach or suggest receiving a plurality of requests from a multi-threaded application. For a detailed discussion regarding Kimmel's (and Maresco's) failure to teach receiving requests from a multi-threaded application, please see the above discussion regarding claim 1.

The Examiner cites column 11, lines 5-30 and element 144a of Fig. 8 in Kimmel. However, the Kimmel reference does not include a FIG. 8 nor an element 144a in any FIG. Since the same passages (column 1, lines 5-30 and element 144a of FIG. 8) where cited in Cheeseman, et al. (U.S. Patent 6,680,933) in the Final Office Action dated December 17, 2004, applicants assume the Examiner mistakenly used the same passages citations in the current rejection of claim 4. Thus, the Examiner has failed to point out or cite any passage of Kimmel that teaches or suggests wherein secondary scheduler is executable to receive the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. Nowhere does Kimmel mention anything regarding a lock through which a plurality of requests may be received in a thread-safe manner from a multi-threaded application.

The Examiner also cites column 2, lines 30-38 and column 6, lines 15-19 of Maresco and argues that Maresco teaches a thread-safe system. However, nowhere, whether at the Examiner's cited passages or elsewhere, does Maresco teach or suggest anything regarding receiving the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. Thus, neither Kimmel nor Maresco, whether considered singly or in combination, teaches or suggests wherein secondary scheduler is executable to receive the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. For at least the reasons above, the rejection of claim 4 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 5, Kimmel in view of Maresco fails to teach or suggest wherein the primary scheduler is executable to receive the plurality of requests from the secondary scheduler through a lock in a thread-safe manner. The Examiner cites column 24, lines 14-26 and column 6, lines 43-65. However, neither of the cited passage mentions anything about Kimmel's dispatcher, which the Examiner equates to a primary scheduler, being executable to receive requests from the medium term scheduler, which the Examiner equates to a secondary scheduler, through a lock in a thread-safe manner. Instead, as noted previously, The Examiner's first cited passage (column 24, lines 14-26) describes how Kimmel's operating system may function in systems having various number of execution queue levels and that have different sized thread groups. The second passage describes the thread group structure maintaining cumulative timeslice and job processor (JP) accounting for all threads in a thread group. For a more detailed discussion regarding these passages, please see the above discussion regarding claim 1. As argued above, regarding claim 1, Kimmel in view of Maresco fails to teach or suggest a secondary scheduler sending requests to a primary scheduler. Additionally, Kimmel fails to mention anything regarding a lock through which a primary scheduler may receive requests from a secondary scheduler.

The Examiner has failed to cite any portion of Maresco regarding the rejection of claim 5. However, Applicants assume the Examiner intended to cite column 2, lines 30-

38 and column 6, lines 15-19 of Maresco. However as noted above regarding claim 4, Maresco fails to teach anything regarding a lock through which requests may be received.

Thus, the Examiner's combination of Kimmel and Maresco fails to teach or suggest wherein the primary scheduler is executable to receive the plurality of requests from the secondary scheduler through a lock in a thread-safe manner. Therefore, the rejection of claim 5 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 16, Kimmel in view of Maresco fails to teach or suggest wherein executing the management requests on the managed objects further comprises sending the management requests to a management information server coupled to the managed objects. The Examiner cites column 12, lines 7-15 and column 11, lines 23-40 of Kimmel. However, the cited passages describe how Kimmel's dispatcher, which the Examiner equates to a primary scheduler, searches for a languishing thread group to execute. No mention is made by Kimmel, either in the cited passages or elsewhere, of a management information server coupled to managed objects. Management information servers and managed objects are well understood in the art and a dispatcher searching through Kimmel's hierarchical data structure to locate a languishing through group to execute does not have any relevance to either a management information server or to managed objects.

The Examiner does not rely upon Maresco to teach, nor does Maresco teach, anything regarding sending management requests to a management information server coupled to managed objects. Thus, the Examiner's combination of Kimmel and Maresco fails to teach or suggest wherein executing the management requests on the managed objects further comprises sending the management requests to a management information server coupled to the managed objects.

For at least the reasons presented above, the rejection of claim 16 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks apply to claim 28.

Regarding claim 17, Kimmel in view of Maresco fails to teach or suggest wherein each of the management requests comprises a corresponding callback function. The Examiner cites column 3, lines 37-48 of Maresco. However, Maresco does not describe anything regarding management request comprising callback functions. The Examiner cited passage describes how Maresco's CWorker object class maintains a reference to the work crew that it belong to and how each CWorker also keeps track of the next and previous members of the worker list for the work crew. Neither Kimmel nor Maresco teaches anything that can be interpreted as management request each comprising a corresponding callback function. Thus, the rejection of claim 17 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks apply to claim 29.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-48600/RCK.

Also enclosed herewith are the following items:

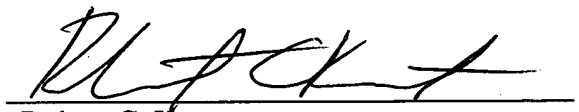
☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: July 6, 2005